In the Claims:

1. (original) A method of automatic configuration of a microprocessor architecture whereby:

(a) the architecture contains a configurable number of execution units;

(b) the architecture has configurable connectivity between those execution units; and

(c) the data and control flows within a particular input program are used to influence decisions regarding execution unit replication and connectivity.

2. (original) The method according to claim 1 whereby multiple candidate architectures are generated.

3. (original) The method according to claim 2 whereby the best candidate architecture is automatically selected on the basis of user defined metrics.

4. (original) The method according to claim 2 whereby data is output to allow the construction of a graph representing the characteristics of certain candidates.

5. (original) The method according to claim 2 whereby a number of new connections and or execution units are added to the architecture on each generation.

6. (original) The method according to claim 5 whereby mapping of code onto a trial architecture is used to influence connectivity choices.

7. (original) The method according to claim 6 whereby the delays caused by execution unit conflicts in the schedule are used to increase the chances of an additional execution unit of that type being added to the architecture.

8. (original) The method according to claim 1 whereby every candidate generated contains a certain minimum set of execution unit types.

9. (original) The method according to claim 8 whereby there is a certain minimum connectivity between all execution units.

10. (original) The method according to claim 9 whereby the minimum connectivity guarantees that arbitrary new code sequences can be mapped to the architecture.

11. (original) The method according to claim 10 whereby the guarantee of mapping of new code is performed using a reachability analysis between results and operands within the architecture.

12. (original) The method according to claim 11 whereby the reachability analysis ensures that the result of every type of execution unit can be transported to the operand of every type of execution unit.

13. (original) The method according to claim 11 whereby the reachability analysis ensures that every result can be written to a central register file and that every operand can be read from a central register file.

14. (original) The method according to claim 1 whereby the initial connectivity within the architecture is determined from data flows within graph representations of the input program.

15. (original) The method according to claim 5 whereby the new connections are added as a result of connections requested during the code generation process.

16. (original) The method according to claim 15 whereby the addition of new connections is constrained by certain rules.

17. (original) The method according to claim 16 whereby the connectivity rules relate to maximum number of operand inputs, maximum number of result outputs and the estimated connectivity distance.

18. (original) The method according to claim 17 whereby a set of potential new connections are maintained and those which are added are those which will improve the fitness metric most but are within the constraints.

19. (original) The method according to claim 1 whereby the execution units are placed in a logical grid layout.

20. (original) The method according to claim 19 whereby the connectivity rules include a maximum distance between execution unit grid positions.

21. (original) The method according to claim 19 whereby the execution units are obtained from a component library, each of which includes a reference to the hardware description of the execution unit.

22. (original) The method according to claim 21 whereby the execution unit components have pre-characterised characteristics that may include area, maximum operational frequency and average power consumption.

23. (original) The method according to claim 22 whereby each architecture is designed to have a minimum operational frequency on a given implementation technology.

24. (original) The method according to claim 19 whereby the placement of execution units is initiated with the register file at the centre of the grid.

25. (original) The method according to claim 24 whereby the placement of execution units is performed from the centre outward in order of decreasing usage frequency of the execution units.

26. (original) The method according to claim 1 whereby the architecture may be optimised for a certain set of functions that are specified to the system.

27. (original) The method according to claim 1 whereby the number and type of execution units and their connectivity for a given architecture is stored in a description file.

28. (original) The method according to claim 27 whereby the stored information includes details of the execution word layout used to control each of the execution units.

29. (original) The method according to claim 28 whereby the stored information includes details of the selection codes associated with operand inputs, output registers and execution unit operation codes.

30. (original) The method according to claim 27 whereby a top level hardware description language file may be generated from the information stored that instantiates connectivity and required execution units.

31. (original) The method according to claim 30 whereby the required package or library files are automatically generated to incorporate the hardware descriptions for all the required execution models for the purposes of hardware synthesis.

32. (original) The method according to claim 31 whereby instruction level profiling information is used to influence the weighting of individual instructions.

33. (original) The method according to claim 32 whereby the weighting of individual instructions during scheduling is related to the profile weighting.

34. (currently amended) A microprocessor automatically configured using a method as defined in Claim 1 ~~any preceding Claim 1 – 33~~.